Introduction

Introduction

Abstract

The nuclear-cytoplasmic ratio (NCR) is a widely used clinical indicator for cancer. The automated extraction of cell size and nuclear size from tissue images is the only practical method to assess quantitatively the NCR in a real-time clinical setting. In this study we demonstrate the feasibility of automatically detecting the cell boundaries and estimating the cell sizes from images of chicken adipose tissue using several image processing techniques and an implementation of the Hough Transform.

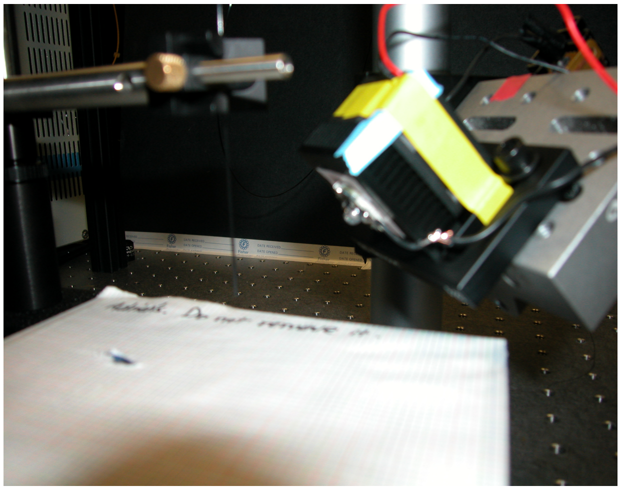
Goals

Using a completely non-invasive image capturing technique, we want to be able to automate a process to detect the cells in the image, as well as calculate the area of the cells. By doing this, when imaging techniques become more advanced, the process can be used to determine the NCR and thus, become an automated, non-invasive indicator of the presence of cancer.

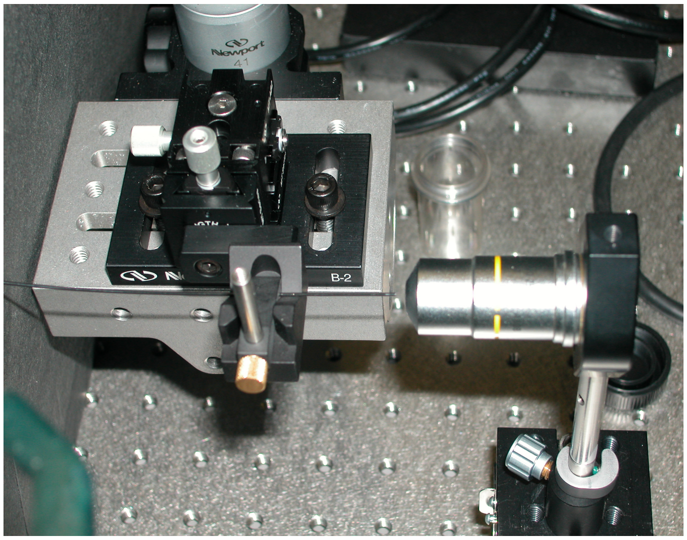
Image Acquisition

The Optical-Fiber System

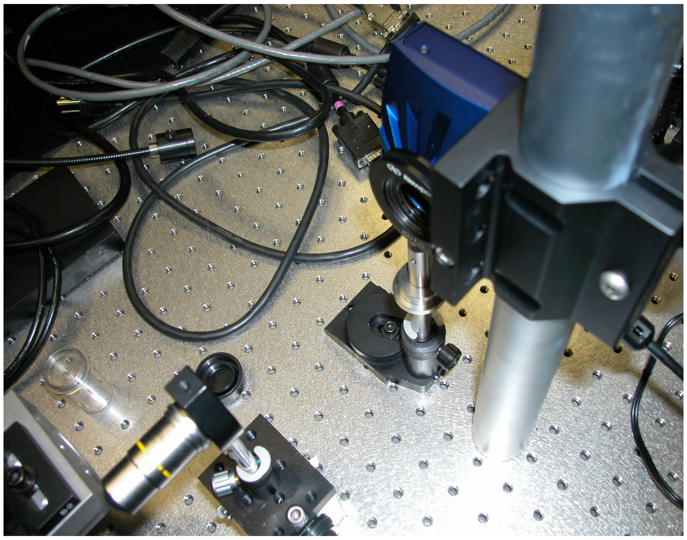
We have designed an optical system to test the imaging capabilities of a one millimeter diameter optical fiber probe. We used a 455 nm Luxeon LED (Philips, San Jose, CA) as the illumination source in accordance with our design goal of a device capable of both reflectance and fluorescence imaging while remaining as simple as possible. A broadband light source would be more ideal for reflectance imaging. However, a broadband light source could perform fluorescence imaging whereas our source would be suitable for both fluorescence and reflectance imaging. The LED was cooled by a heat sink and fan assembly mounted on its back and powered by a 12V AC/DC adaptor. To mimic the fatty environment of the human breast, test images were taken from samples of chicken adipose tissue.



The 1-mm diameter fiber (Sumitomo Electric USA, Los Angeles, CA) was positioned next to the LED and over a vertically adjustable sample stage. (Figure 2) The fiber tip was adjusted to receive maximum illumination from the LED when it was in contact with the sample. The fiber collected the light reflected by the sample at its distal end and conveyed it to its proximal end. The proximal end of the fiber was focused at the focal plane of an infinity corrected microscope objective (Newport, Irvine, CA). The position of the fiber was adjusted by a three-axis micro translation stage (Newport). The infinity corrected object could project an image from its rear that is in focus regardless of the distance between the object and the detector.



The detector was a Zeiss MRc5 5.0 megapixels color CCD camera (Carl Zeiss, Thornwood, NY). The magnification of the image on the detector relative to the image on the proximal fiber tip was the ratio between the distance from the detector to the back of the objective and the distance from the front of the objective to the fiber proximal tip. The infinity corrected objective permitted a wide range of magnifications (a while range of allowable detector-objective separation) while maintaining the focus of the image. (Figure 3) This feature could potentially allow the incorporation of a movable detector that can change the magnification of the image on command.



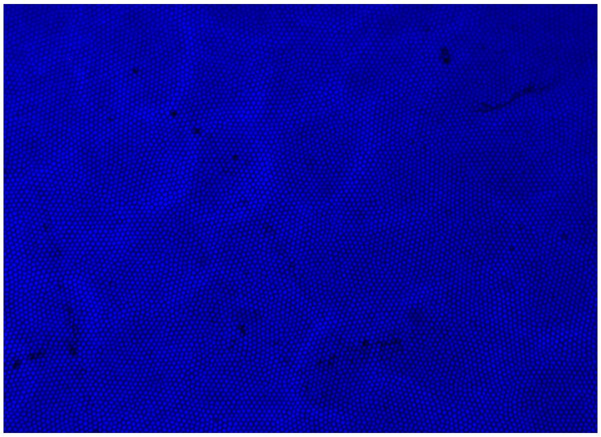
The image from the detector was captured onto a laptop computer (Lenovo, China) with the Zeiss AxioVision camera software. (Figure 4) The image was exported as a bitmap file. The bitmap file was subsequently processed in Matlab (Mathworks, Natick, MA).

Defining Borders

Defining Borders

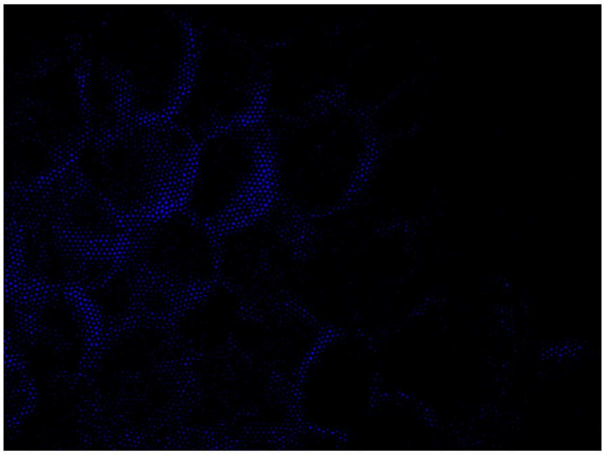
Overview

With the original cell picture, as shown below, it is very difficult to make out the cell borders due to the lack of contrast between the borders and the background of the image. Our first goal, before edge detection can even be done, is to create a sharper contrast between the borders and the background. This is shown in Figure 1.



Thresholding

In order to create a higher contrast between a cell’s borders and the background, thresholding was utilized. Pixels of lower intensities, the background, were thrown out while pixels of higher intensities were kept. The IMADJUST command in Matlab allowed us to specify a threshold intensity and mapped all pixels below the threshold to zero. The result can be seen in Figure 2.



Filtering

Overview

Although we have more contrast between our cell borders and the background, there is still a lot of noise which will need to be annihilated before edge detection can be successful. Ideally, we want to only preserve pixels which constitute a cell border and map all other pixels to a zero intensity.

Conversion to Gray Scale

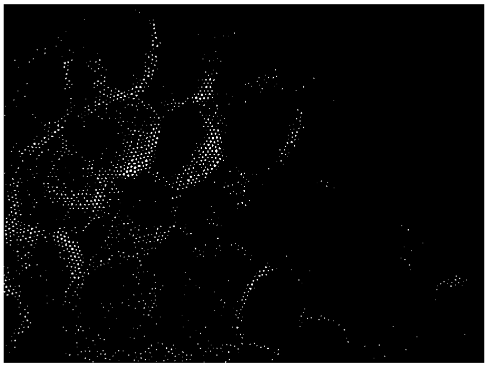
Filtering is more easily done on a grayscale image than on a colormap image. This is due to the fact that grayscale images are intensity images with each pixel reflecting a particular intensity, rather than a combination of three colors (red, green and blue). With a grayscale image, filters such as mean and mode filters can be more appropriately applied.

Mean Filtering

The first step in getting rid of unwanted pixels is to apply a mean filter. This filter will take a specified neighborhood around each pixel and set each pixel to the mean of its neighborhood. This is done with Matlab’s sliding neighborhood operation which will go through each pixel, taking its neighborhood pixels as inputs into a specified function and set each pixel as the output of the function.

In our case, the function we want to apply will take the neighboring pixels and output the mean. This mean represents the average intensity of its neighboring pixels. Thus, a lone intense (lit) pixel way out in the boondocks will be set to zero since there are no other high-intensity pixels nearby. This will effectively eliminate lone high-intensity pixels and thus decrease noise.

Following mean filtering, the image was converted to black and white. This was done by specifying a threshold intensity and setting all pixels above the threshold to 1 (white) and all pixels below to 0 (black). Why must this be done? Well, you’ll just have to read on to see… Figure 1 shows our final result in this step after converting to black and white.



Connecting Lines

Connecting the Lines

Rationale

OK, so now we have a (relatively) clean image of the cell’s borders. Can we run an edge detector now you ask…? Hold your horses, Champ, we still have a ways to go…

Although it may make sense to run an edge detector at this point, seeing as we’ve got ourselves an image where the cell borders are definitely distinguishable, there is still a problem. Edge detectors look for changes in the gradient. Since our image was acquired utilizing optical fibers, our cell borders are not really solid lines quite yet; they are merely groups of small dots which, together, make up the cell borders. If an edge detector were utilized at this point, it would pick up each fiber optic probe, rather than the cell border we want. There no need to despair; there is a solution for all this!

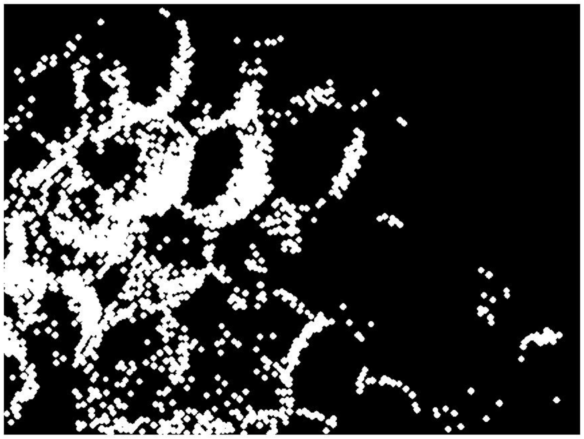
Mode Filtering

Although it may not be as simple as those connect-the-dots books we’re used to, mode filtering can be an effective method of forming solid cell borders. The basic idea is this: if we used sliding neighborhood operations again, but rather than look for the mean, look for the mode, we may be able to connect all our little dots together! Why is this? Well, the mode filter we implemented works like this:

1. Take a neighborhood of size [N M]
2. Find the mean of the entries in the matrix.
3. Since the image is in black and white, if the mean is greater than .5, then there are more ones, otherwise, there are more zeros. This is basically determining the mode of the neighborhood.

4. Thus, if the mode is 1, set the pixel to 1. If the mode is 0, set the pixel to 0.

How does this connect the lines? Think about this: we want to establish solid cell borders by turning ON the black pixels which are part of the cell's border. Thus, we want to turn ON black pixels which are near groups of white pixels which make up the cell borders and keep black pixels which are not part of the cell borders (in areas with relatively few white pixels) OFF. Taking the mode of each pixel's neighborhood will accomplish this since black pixels near large groups of white pixels will be turned on and the following image shows what begins to look like solid cell borders… (Figure 1)



Edge Detection

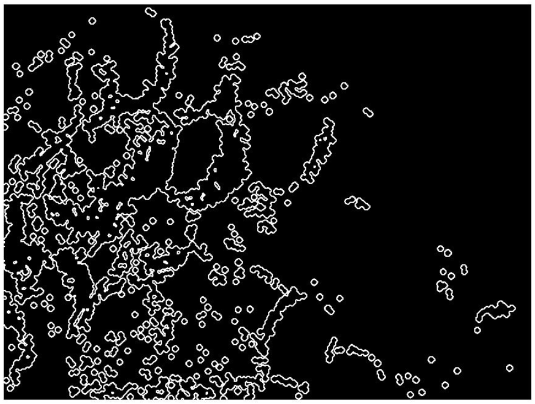
Canny Edge Detector

We use the Canny Edge Detector to (hopefully) obtain sharp edges for our image now that we have “blurred” together the dots which form the cell edges. The Canny edge detector first takes a gradient of the image. A gradient magnitude, as well as the edge direction (which way the gradient is increasing) can then be found. The Canny edge detector does this by using four different filters which can detect horizontal, vertical and diagonal edges. For each pixel, the filter which outputs the largest response can be found and the direction determined.

Distance Matrix

Once we have run our image through the Canny edge detector, a distance filter is used to connect all remaining lines in a cell’s border. The distance function calculates the Euclidean distance between each pixel and the nearest pixel which is on. The output of the distance function is a matrix having the same dimensions as the image matrix. The entries correspond to the distances; for example, if pixel (i,j) was a distance x away from the nearest on pixel, the (i,j) entry in the distance matrix will be x.

After the distance matrix is found, an iterative method is used to go through each entry of the matrix. A threshold distance is determined and every entry of the distance matrix which falls under the threshold is mapped to one, and every entry which falls above is mapped to zero. What exactly is this doing? Well, if a pixel is close enough to an on pixel, then we assume it is part of the cell’s border, and thus must be turned on as well. The distance function and subsequent thresholding is performed multiple times, with the threshold getting small and small each time until the cell borders are completely continuous. The result is shown below.



Hough Transform

And now, Introducing the Hough Transform…

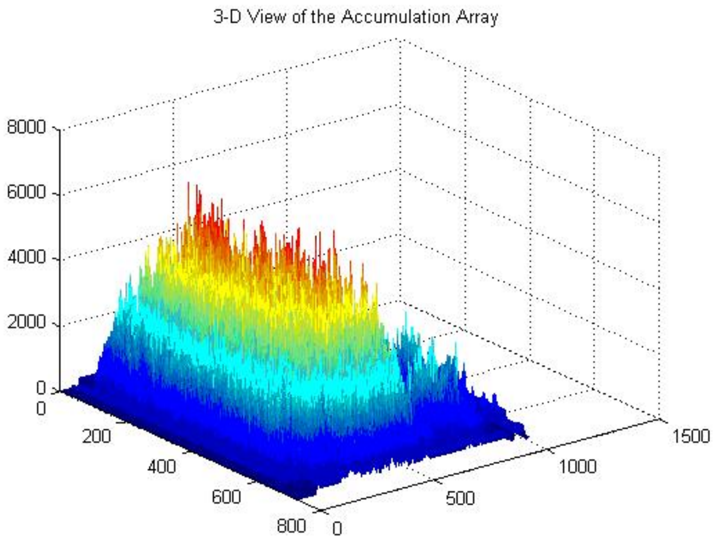
We are finally about to get to the fun part: circle detection…

The Hough Transform

The Hough Transform is performed to recognize circular patterns based on parameters such as circle radius range, gradient threshold and presence of concentric circles. The Hough Transform will in as an input a range of radii. It will then generate circles of varying radii which cover the entire range. These circles are convolved with the image and the goodness of match is recorded in an accumulation array.

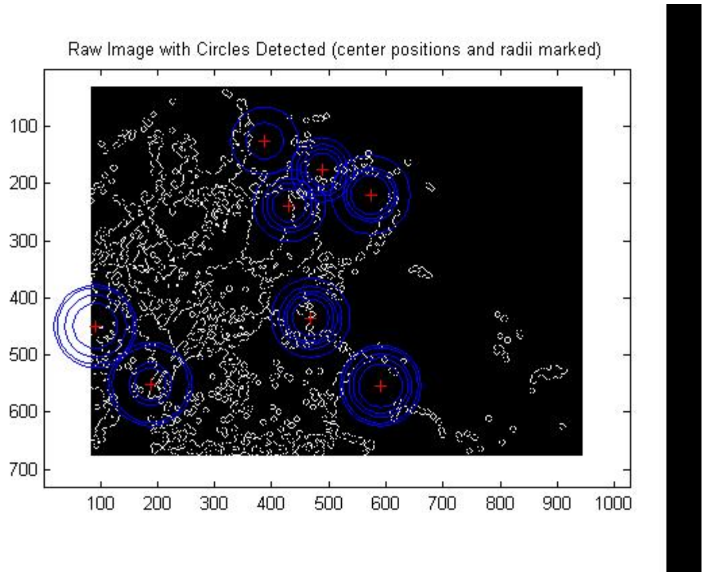
Note: Generally, convolution is not performed, but rather, the FFTs are taken and multiplied together. This is not only more efficient, but more easily implemented as well!

We found a nifty Matlab function which implements the transform and ran it on our image. The accumulation graph can be seen below, with the peaks corresponding to the best circle matches.



Circles, Circles, Circles!

With the circles detected, the areas where the best circular matches were can be drawn out in our original image. The result is quite striking: of the twelve or so cells which could be picked out by the human eye, the Hough Transform was able to detect roughly eight cells.



Conclusion

Conclusion

Area Calculation

With the accumulation array, we were able to extrapolate the sizes of the individual cells and it was determined that the average cell size in our image is roughly 150um2.

But what does it all mean?

When imaging techniques become more advanced, it is our hope that not only would we be able to calculate the size of the cells, but the nucleus as well. By being able to calculate both the cell sizes as well as the nucleus, the ratio between the two can be determined. This can then be used as a completely non-invasive detector of cancer, with higher nucleus-to-cell ratios being indicative of the presence of cancer.

Improvements

Our area calculation is far from perfect and, with time, there are many improvements which can be made.

Image Acquisition

The image acquisition technique can be improved to not only be able to detect the nucleus, but to provide better definition between a cell’s border and the background as well. But… this we will leave to our bioengineering friends.

Edge Detection

In our edge detection phase, one problem we faced was the varying thickness of cell borders. Sometimes, especially thick borders would appear as a closed region resembling a cell when edge detection is performed! This problem, however, we believe can be rectified using carefully constructed filters which can hopefully reduce the thickness of borders.

Circle Detection

The Hough Transform we used approximates cell sizes using the approximate circles. Cells, however, are not always completely circular. A more generalized transform can be employed to more precisely detect the shape of the cells and give a more precise area calculation.